Multiple Signature Algorithms and the Bridge CA Concept

Bill Burr

301-975-2914

william.burr@nist.gov

July 9, 1998

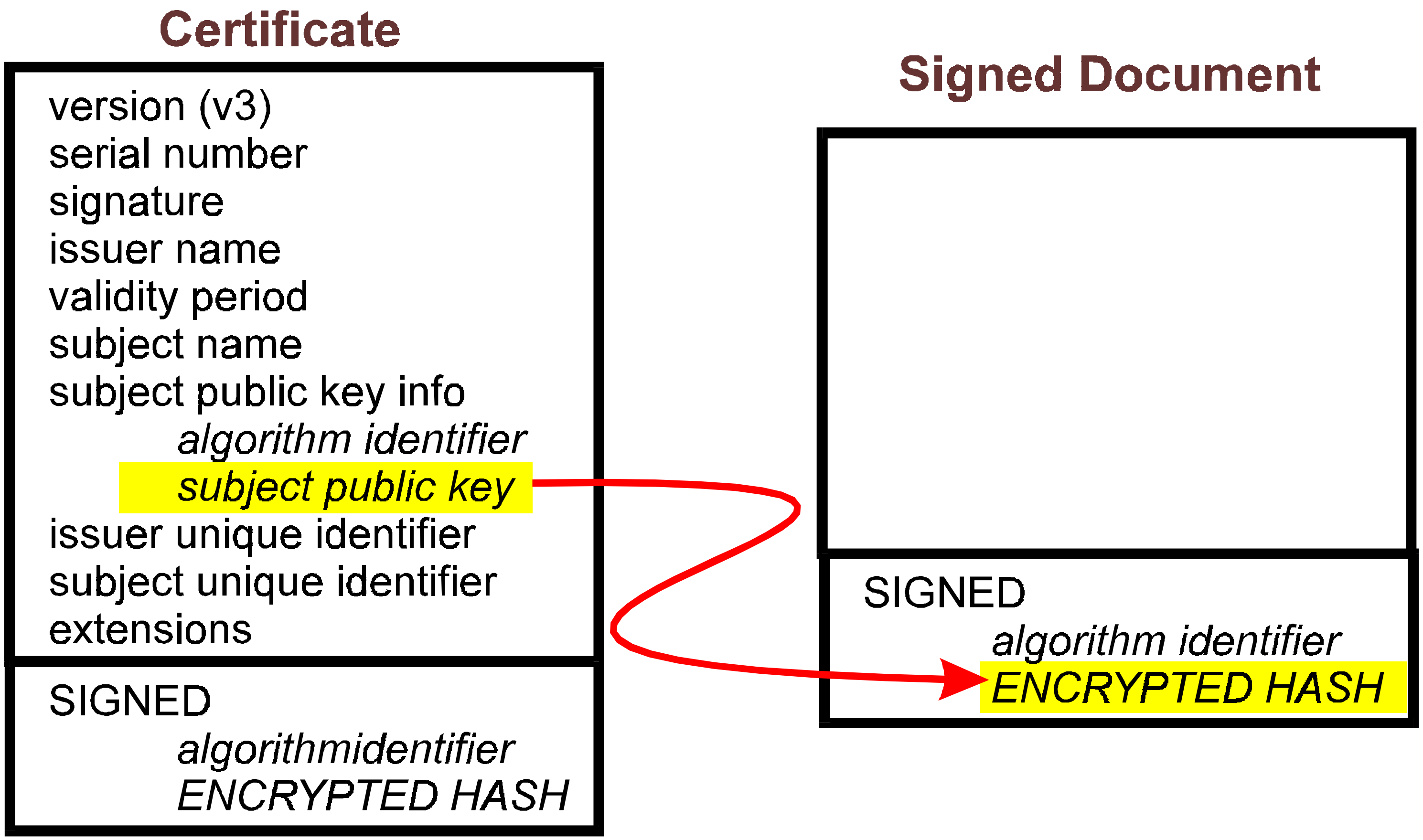
Current Federal Situation

- **Numerous Federal PKI pilots**
  - – built and paid for for some agency application
    - justified in terms of benefit to that application
- **Different Architectures**
  - – mesh (Entrust), browser (DoD, ACES, etc.), &Hierarchical (MISSI-DMS)
- **Different Algorithms**
  - – DSA, RSA and, soon, EC-DSA

Current Situation

- **Little interoperability between pilots**
  - At present interoperability is a hard problem at the practical level
  - Has been more difficult than you would think even to achieve cert. path interoperation between CAs from the same vendor.
- **Multiple algorithms make the problem worse**

Certificate and Signed Document



Certificate

version (v3)
serial number
signature
issuer name
validity period
subject name
subject public key info
algorithm identifier
subject public key
issuer unique identifier
subject unique identifier
extensions

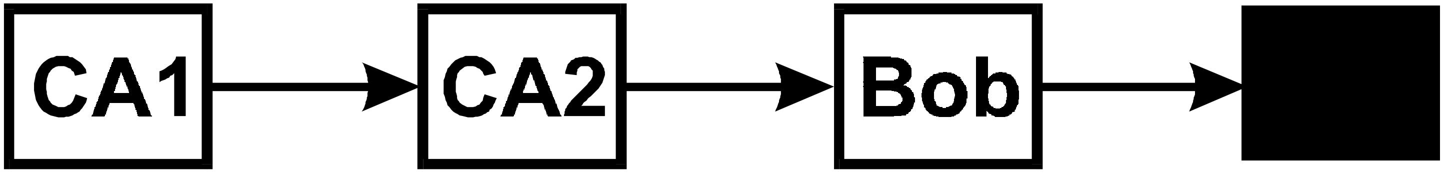
SIGNED
algorithmidentifier
ENCRYPTED HASH

Signed Document

SIGNED
algorithm identifier
ENCRYPTED HASH

Certification Path

- **Alice verifies Bob's certificate by verifying a *certification path* ending in one issued by a CA she trusts**



Alice trusts CA1

Certification Path Interoperability

- **Primary interoperability issue is can Alice find and process a certification path to Bob, when they have different CAs?**
- **Many other CA to CA cross-certification, CA to repository, repository to repository, CA to RA interoperability issues**

Digital Signature Algorithms

- **Several digital signature algorithms in use**
  - RSA
  - DSA
    - parameters
  - ECDSA
    - parameters

Parameters

- **Publicly known constants**
  - – usually the same for all certs. issued by a CA
  - – can be big numbers
    - same general size as public key
- **Specified in subjectPublicKeyInfo field of certificate**

Parameter Inheritance

- **Makes certificates smaller**
- **If parameters aren’t specified in publicKeyInfo field, they are “inherited” from previous step in certification path**

Parameter Inheritance

- **Not specified in X.509**
  - – incorporated in PKIX
  - – done in MISSI
    - only “root” and high level (“PAA”) CAs normally include parameters in their certificates; subordinate CAs and end-entity certificates inherit their parameters

Definitions

- **Consistent certificate**
  - – subject and signer algorithms are the same
  - – parameters can be inherited
- **Hybrid certificate**
  - – subject key and signer algorithms are different
  - – allowed by X.509
  - – subject parameters must be specified
  - – relying party must validate 2 algorithms

Hybrid Certificates

- **Must have one in path if Bob and Alice use different algorithm**
- **Otherwise are undesirable**
  - – need to implement 2 algorithms to use them
  - – may be large, because of parameters
- **Goals:**
  - – never have more than one hybrid in cert. path
    - never introduce 3rd algorithm in path

Interoperability Approaches

- **Parallel PKIs**
  - – separate PKI for each algorithm
    - expensive
    - no hybrid certificates
  - – user has certificates (and perhaps clients) for each algorithm needed for interoperability
    - how many certificates does he need?
    - how many can he manage?
    - simpler (but perhaps more) clients

Interoperability Approaches

- **End-Entity**
  - clients may sign with only one algorithm, but are expected to validate all algorithms
    - user needs only one certificate
    - some extra expense in clients
    - inconsistent certificates are needed for interoperability

Hybrid Certificates

- **Hybrid end-entity certificates usually make little sense**
  - every relying party must be able to validate both algorithms
  - even certificate holders of the same CAs must validate 2 algorithms to interoperate
  - requires parameters be specified in end-entity certificates

General Approach

- **End-entity solution is best**
- **Use consistent end-entity certificates**
- **Consistent trust domains desirable**
  - minimize interop problems in domain
- **One signature algorithm per CA**
  - a CA is just a name in this context
    - create a new name for each algorithm
    - avoids mixed algorithm Certificate Revocation Lists

Parameters

- **Specify parameters only**
  - in self-signed certificates
  - in hybrid certificates
  - when the parameters for the subject key are different than the signing key

Bridge CA Approach

- **Build nexus to connect the pieces**
- **Three key elements:**
  - – Federal Policy Management Authority (PMA)
  - – Federal "Bridge" CA (BCA)
    - not a root
    - cross certifies with CAs
  - – Bridge CA Repository
    - for CA certificates and status

Federal PMA

- **Overall management of FPKI**
- **Supervises BCA and BCA Repository**
- **Sets overall Federal Cert. Policies**
  - assurance levels
  - model policies
- **Approves Bridge CA cross-certification**
  - reviews CA CPS

Trust Domain

- **A group of CAs that**
  - – operate under the supervision of a Domain Policy Management Authority
  - – use consistent policies, and have similar Certification Practice Statements (CPS)

Bridge CA (BCA)

- **Cross certifies with “Principal CA (PCA)” in each trust domain**
  - ***not a root***: does not start cert paths
  - may have constraints in the certs it issues
  - also cross certifies with non-Federal PCAs
- **Issues Authority CRL (ARL)**
  - CRL for all Federal CAs (and perhaps others)
  - Modest size, since CA certs. are not volatile

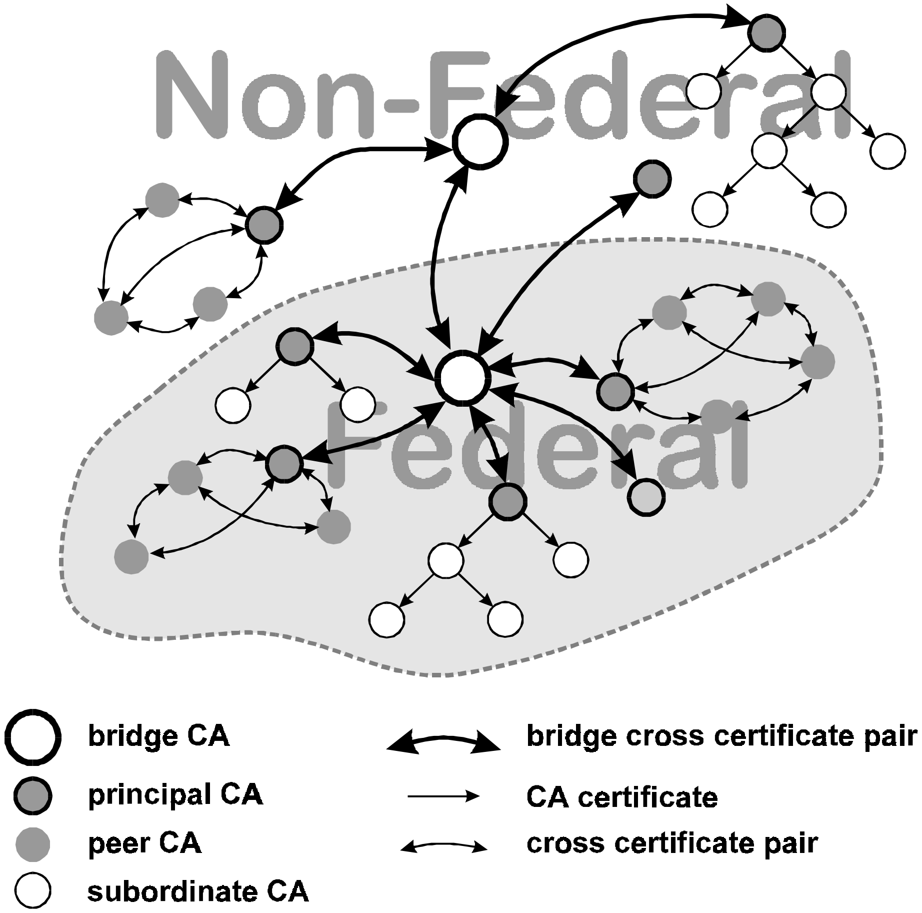
Bridge CA Repository

- **One-stop shopping for CA certs.**
  - – CA certs. for the Federal PKI
  - – ARL
- **High availability**
  - – key to building cert. paths
- **Medium bandwidth**
  - – everything it holds can be cached
  - – ARL should not be large

Principal CA

- **Designated CA in each trust domain**
- **Has cert. path to all other CAs in the domain**
- **In hierarchical domain, the root CA**

Bridge CA FPKI Architecture

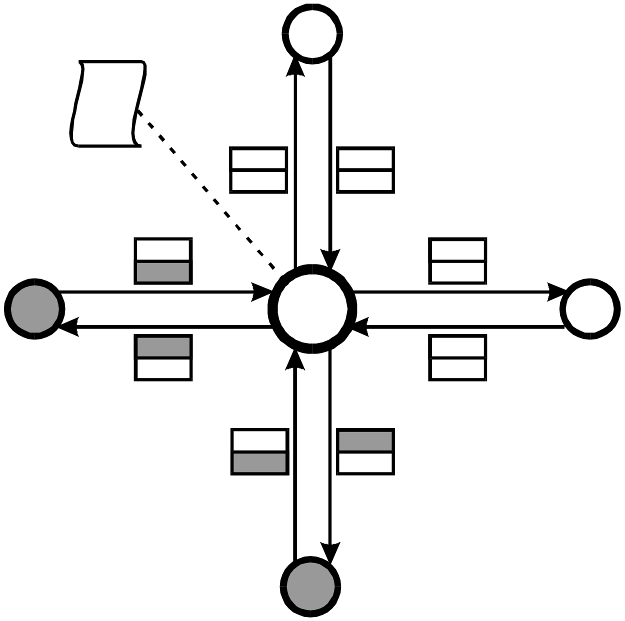


Possible BCA Approaches

- **Preferred algorithm**
- **Multiple algorithm bridge**
- **Split bridge**

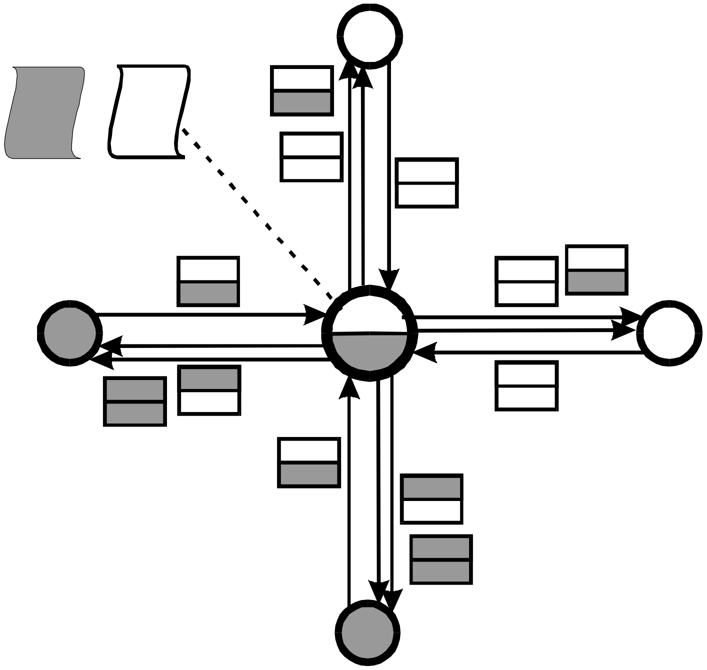
Preferred Algorithm Approach

- **Bridge signs with one algorithm**
  - everybody who uses BCA must validate this algorithm
- **Efficient**
- **Can we pick one algorithm and make it stick?**



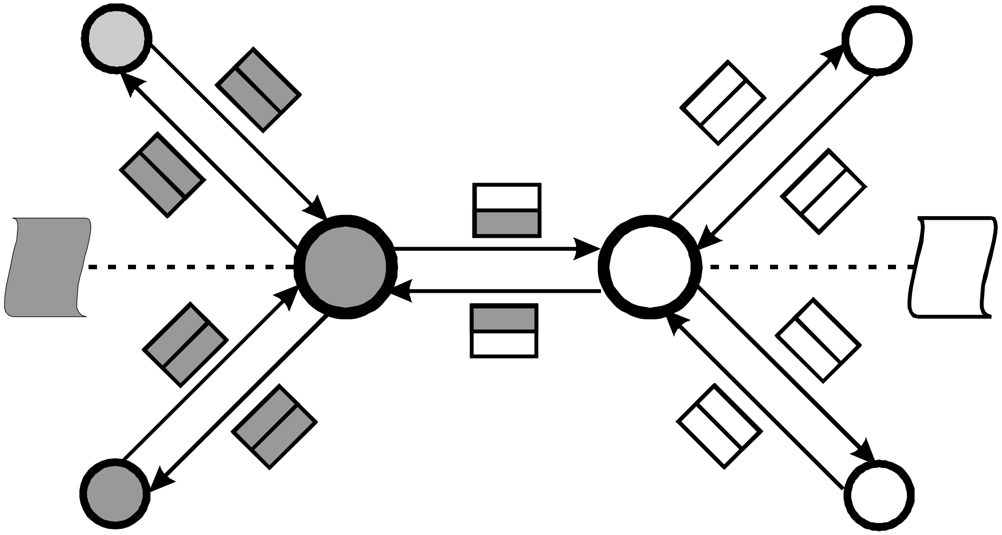
Multiple Algorithm BCA

- **BCA signs with several algorithms**
  - issues all hybrid certificates to PCAs
- **BCA issues several ARLs**
  - one per algorithm
- **To make cert. path, how do we easily identify needed PCA certificates?**
  - several for each PCA



Split Bridge CA

- **Separate Bridge CA per algorithm**
  - each BCA has a separate name, by not necessarily a separate physical workstation



Split Bridge CA

- **All hybrid certs occur between BCAs**
- **Fewer additional hybrid certs than Multiple Algorithm Bridge**
- **Separate BCA names may simplify finding the right hybrid cert or ARL**
- **Hybrid cert becomes an extra step in cert paths**

Conclusion

- **Bridge is the right point to provide hybrid certs to address multi-algorithm interoperability**
- **Question: which BCA oriented approach do we prefer?**